NUMERICAL SIMULATION OF PARTICLE TRANSPORT IN A DRIFT RATCHET

MARKUS BRENK, HANS-JOACHIM BUNGARTZ, MIRIAM MEHL, IOAN L. MUNTEAN, TOBIAS NECKEL, TOBIAS WEINZIERL *

Abstract. The directed transport of micro-particles depending on their size is the basis for particle sorting methods that are of utmost importance in life sciences, e.g. A drift ratchet is a Brownian motor that allows for such a directed transport. Hereby, the particle motion is induced by a combination of the Brownian motion and asymmetries stemming for example from the domain's geometry, electrical fields, or transient pressure boundary conditions. We simulate a particular drift ratchet which consists of a matrix of pores with asymmetrically oscillating diameter wherein a fluid with suspended particles is pumped forward and backward, and where the particles' long-term transport direction depends on their size. Thus, this setup allows for the continuous and parallel particle separation, which has been shown experimentally. However, for a deeper understanding and for an optimized parameters' choice further investigations, i.e. simulations, are necessary.

In this paper, we present first results achieved with our parallel three-dimensional simulation codes applied to a still simplified scenario. This simplification is necessary to isolate different phenomena (asymmetries and Brownian motion, e.g.) in order to check their relevance for the particle transport. The simulation codes are based on (adaptive) Cartesian grids in combination with finite volume and finite element discretisations. Cartesian grids allow for a very efficient implementation of the solver algorithms and an efficient balanced parallelisation via domain decomposition. The achieved simulation results show the effectiveness of our approach and give some strong hints on a directed particle transport already with the simplified model we used here.

Key words. particle sorting, drift ratchet, Brownian motion, fluid dynamics, Eulerian approach, Cartesian grid

AMS subject classifications. 35-04, 35Q30, 65-04, 65M60, 68-04, 68P05, 68U20, 68W10, 70-04, 70E99, 76-04, 76D05, 76M10, 76M12

1. Introduction. In life sciences, methods for sorting macromolecules such as proteins or DNA according to their size are of utmost importance. However, there are no generally applicable methods known yet. For example, gel-electrophoresis [31, 33] as the established and commonly used method for DNA sorting strongly depends on a deeper knowledge of the chemical and physical properties of the DNA particles. Thus, methods using micro-lithographically produced "sieves" or "channels" have been developed as an alternative approach. In this context, the principle of Brownian motors ("ratchets") [1, 29, 30] with electrical fields acting as driving forces for particle motion has been examined but has not yet provided technically utilisable outcomes. These Brownian motors are based on the fact that thermal fluctuations and asymmetries in the applied electrical fields induce a particle transport in non-equilibrium systems, and the long-term direction depends on particle properties such as the particle size.

In contrast to these sieve and channel systems, the required asymmetries in our drift ratchets [21] are included in the channels' geometry. These channels – called pores – have a radius varying with the axis direction. In addition, the drift ratchets use

*Institut für Informatik, TU München, Boltzmannstraße 3, 85748 Garching, Germany ({brenk,bungartz,mehl,muntean,neckel,weinzier}@in.tum.de).

an oscillating flow of a fluid in which the particles are suspended as a stimulus for particle motion instead of electrical fields [20, 25]. The core of our ratchets is a matrix consisting of a high number of the three-dimensional pores (see Fig. 1.1). This matrix is embedded in a fluid domain with one fluid basin at each side of the pore matrix. At the beginning of the sorting process, particles are evenly suspended in the fluid. A pressure pump induces an oscillating forward and backward flow through the pore matrix. A directed movement of particles is induced by asymmetries in the pore geometry (see Fig. 1.1), by the oscillations of the applied pressure [21], and by the Brownian motion of the fluid and, thus, of the particles, too. As the direction of the particle movement depends on the particle size, this method can be used for a *continuous* and *parallel* particle sorting.

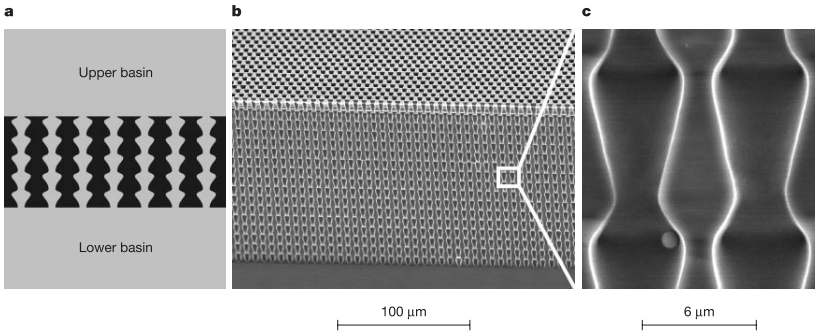


FIG. 1.1. *Visualisation of the drift ratchet taken from [21]. Schematic representation (***a***), scanning electron micrograph of the complete porous structure (***b***) and of two pores (***c***). The length of one pore period is* $8.4\mu m$*, the minimum diameter is* $2.5\mu m$*, and the maximum diameter is* $4.8\mu m$*.*

The long-term objectives of our work are the explanation and understanding of all relevant physical effects, the identification of the main influencing factors for particle transport direction, and, based on this, the derivation of a simplified mathematical model that is still reflecting all relevant phenomena with a sufficient accuracy. A question in this context is, for example, whether the full coupling between fluid flow, particle transport, and Brownian motion can (partly) be neglected or not. The resulting "minimised" model will, finally, be used for the optimization of system parameters (pore geometry, pressure, e.g.) for given sizes of particles to be sorted. The first and essential steps for the isolated examination of the phenomena involved and their relevance for the transport effects are examinations of the transport of a single particle in one geometric period of a realistic three-dimensional ratchet pore. The aims are, first, to build up a data basis in order to judge the applicability of simplified particle transport models such as the Faxen correction [13] that compute the particle movements on the basis of a particle-free velocity field and, second, to observe whether a particle transport can already be observed without taking into account the fluid's Brownian motion. The model we use for this purpose is a combination of the Navier-Stokes equations for the fluid flow and transport equations derived from Newton's law of motion for the suspended particles. As the size of our pores and particles is in the range of micrometers and the particles are considered as non-deformable spherical bodies, we do not have to switch to simulations of molecular dynamics (such as those described in [36, 23]) which would be neccessary on an even smaller spatial scale (nanoscale). An alternative approach for the simulation of the fluid dynamics would be a Lattice-Boltzmann method. With the help of the Chapman-Enskog pro-

cedure, the Navier-Stokes equations can be derived from suitable Lattice-Boltzmann equations. In this sense, both models are equivalent.

Computationally, the resulting coupled low-transport scenario represents a challenging task for several reasons: First, as can be seen from Fig. 1.1, the three-dimensional geometry is rather complicated and has to be represented with a good accuracy which requires a high resolution of the computational grid. Second, we talk about fluid flow with a very low Reynolds number (≤ 0.1), which causes numerical stability problems for explicit time discretisation schemes and, at first sight, suggests to either perform a great many of time steps or switch to implicit time discretisations. Both approaches result in very high computational costs. Third, even if we can solve this instability problem, we still have to compute a big number of time steps as we have to cover several pumping periods of the pressure pump. If we restricted ourselves to smaller time intervals, a directed particle transport would not be observable. Fourth, we examine a coupled multi-physics and multi-scale problem composed of the flow field, the particle movements, particle interactions with the pore's walls, and the Brownian motion. Thus, we are faced with different phenomena in different domains (fluid flow – particle movement), different types of models (deterministic fluid flow – stochastic Brownian motion), and highly different time scales (Brownian motion, pressure frequency and long-term averaged direction of particle movements). Fifth, we have to deal with large particle movements covering the whole computational domain which makes for example Lagrangian grid methods inapplicable. The question is, how to algorithmically realise and implement such a system in an efficient way.

As a general concept to handle the multi-physics problem, we apply a partitioned approach, that is we do not establish and solve an overall equation for flow and particle transport on the whole computational domain but, instead, compute fluid flow and particle movement separately and interconnect both phenomena via suitable coupling strategies (Sect. 2). As underlying models for fluid flow and particle movement, we use the incompressible time-dependent Navier-Stokes equations and Newton's laws of motion (Sect. 3). To discretise our model, we use an (adaptive) Cartesian grid in the fluid domain and a surface triangulation for the particles (Sect. 4.1). In Sect. 4.2, we describe the applied finite volume and finite element discretisations of the Navier-Stokes equations. The time discretisation scheme is described in Sect. 4.3. It includes a solution for the time-scale problem due to the different time scales of the relaxation of fluid flow with its very low Reynolds number as a reaction on changes in the boundary condition on the one hand and the oscillation of the pressure boundary conditions on the other hand. In addition, we address the question how to ensure divergence free velocity fields also in the case of changes of the particle representation in the Eulerian (i.e. fixed) grid and changes of the boundary conditions on the particles' surface as a result of the solution of the motion equations for the particle. Sect. 5 shortly introduces the codes we use. Simulation results are presented in Sect. 6. Finally, we draw conclusions and give a short outlook on our future work in Sect. 7.

2. The partitioned approach. Our scenario – with a fluid phase and explicitly resolved particles transported by the fluid – is a typical multi-physics problem. As the advances both in computer architecture and in numerical methods and algorithms more and more allow for the simulation of complicated scenarios, multi-physics problems have become a central subject in Scientific Computing in the last years. Numerous examples can be found for instance in the fields of fluid-structure interaction

[16, 38, 7] and micro-electronics [15]. Two general approaches are known to handle the resulting models composed of several parts stemming from the different physical phenomena involved: First, the monolithic approach, and, second, the partitioned approach.

The monolithic approach summarises all physical effects in one single model, that is a set of equations covering the whole computational domain and all phenomena to be simulated. The result is a very accurate but also very complex and, thus, computationally expensive model. In many cases, this complexity requires the reduction to a macro model – possibly giving up the resolution of space. In addition, the monolithic model is restricted to a single type of application and can hardly be reused in similar or different cases. In contrast to this, partitioned approaches [27, 28] resort to existing models and software for the single phenomena involved (fluid flow, structure movement, e.g.) and establish a coupling between these submodels via boundary conditions and iteration methods in the final algorithm. We use the partitioned approach due to its simplicity and modularity. Our work is concerned with the derivation of an as simple as possible model for the drift ratchet. Thus, we have to be able to change the setup by either changing single components (fluid flow with and without Brownian motion, particle movement purely induced by the fluid or additionally by an 'own' Brownian motion, e.g.) or adding/eliminating whole components (particles, e.g.). In a monolithic approach, all these changes would require a change of the model, whereas in a partitioned approach, the changes are restricted to the concerned component and, therefore, of course much easier to be realised. For these reasons, we choose the partitioned approach consisting of a fluid and one (or several) particle component(s). Thus, we can integrate the two software components in a modular way and easily enhance the system by additional aspects or exchange components.

To really exploit the whole flexibility offered by partitioned approaches, the implementation of the resulting coupled simulation has to be done very carefully. In particular, a change or exchange of one component of the overall setting should not affect the other components. In most common approaches, the involved simulation codes are directly coupled to each other. That is, they directly exchange data and the iteration process used for the overall time stepping of the coupled system is defined in these codes themselves. Thus, all involved simulation codes have to be adapted to their 'partner' codes and to the coupling strategy (i.e. the type of iteration for time stepping) whenever one component is altered. Even if we use sophisticated and powerful toolboxes such as `MpCCI` [14], we can't completely eliminate these drawbacks: The dependency of all codes on the coupling strategy remains unchanged, and even the data exchange, that is the type of interpolations and projections of data between the computational grids of the involved codes, has often to be tailored to particular features of the solvers such as higher order elements.

To improve this unsatisfactory situation, we have developed the coupling environment `FSI❄ce` [4]. The underlying idea is to use a client-server approach with a coupling software acting as the client which collects data from and sends data and jobs to the solvers that act as servers in this context. The strict separation and hiding of the solvers from each other and from the coupling strategy is guaranteed

- on the algorithmic level by the integration of the whole control of the coupled simulation in the coupling client and
- on the data level by an independent data structure describing the coupling

surfaces (surface triangulation) in the coupling client.

For our drift ratchet scenario, however, the situation is simpler than in the general case. The particles are rigid spherical bodies. Thus, their motion can be described by the four parameters velocities (three in 3D) and rotation speed. The other way round, all the particle needs to know from the fluid is the integrated force acting on its surface. This makes the exchange of grid data at the interfaces between fluid and particles obsolete. Therefore, we use a simplified coupling approach here. The triangulation of the particles' surfaces is only used to establish the initial geometry for the simulation (see Sect. 4.1).

3. Model equations. This section shortly describes the underlying mathematical models both for fluid flow and for particle transport.

3.1. Fluid flow: the incompressible Navier-Stokes equations. As the fluid in the drift ratchet pores is a liquid under moderate pressure without any external forces, we use the three-dimensional time-dependent incompressible Navier-Stokes equations:

$$\nabla \cdot \mathbf{u} = 0, \tag{3.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} - \frac{1}{Re}\Delta \mathbf{u} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} + \nabla p = 0, \tag{3.2}$$

where $\mathbf{u}$ denotes the three-dimensional velocity vector, p the scalar pressure, and Re the Reynolds number of the flow field. On our scale of observation (micro scale), the suspended particles are explicitly considered which makes a non-Newtonian representation of the fluid obsolete.

3.2. Particle movements: Newton's law of motion. Since we consider a particle in the drift ratchet to be spherical, solid, homogeneous, and of fix mass m, the particle motion may be described by a translational and rotational acceleration according to Newton's second law of motion. Thus, we get the simple set of ordinary differential equations

$$\dot{u}_{\mathrm{p}} = \frac{F}{m}, \tag{3.3}$$

$$\dot{\omega} = \frac{M}{\Theta}, \tag{3.4}$$

that allows us to compute the translational and angular velocities u_{p} and ω as well as, by integrating once more for a given time step size dt, the new position x_{bary} of the barycentre and the new angle φ. F denotes the total force on the particle, M corresponds to the total torque acting on the particle. The constant inertia Θ of a homogeneous sphere rotating for example around the x_3-axis is computed as

$$\Theta = \rho \int_V \left(x_1^2 + x_2^2\right) dx = \frac{2}{5} m r^2,$$

where r is the radius of the sphere.

4. Discretisation. This section describes the discretisation of the coupled problem. As already mentioned above, no extra spatial discretisation will be used for the particles (besides from their representation in the computational grid of the flow solver), as their rigid body motion is defined by a small number of scalar values without spatial resolution. Thus, we will restrict to the spatial descretisation of the fluid domain and the flow equations in Sect. 4.1 and Sect. 4.2. For the time stepping (Sect. 4.3), however, the particles' movements play a non-negligible role.

4.1. Grids. As computational grids, we use Cartesian grids in our flow solvers. At first sight, they have the disadvantage of a reduced accuracy in comparison to (unstructured) triangular grids for complicated geometries. But, due to their strict structure and inherent recursive construction (see Sect. 5), these grids offer a lot of advantages in terms of simplicity, runtime, and memory efficiency.

In addition, the lack of accuracy can be remedied using adaptive refinements (see Fig. 4.1) or particular operator discretisations [2] at complicated boundaries.

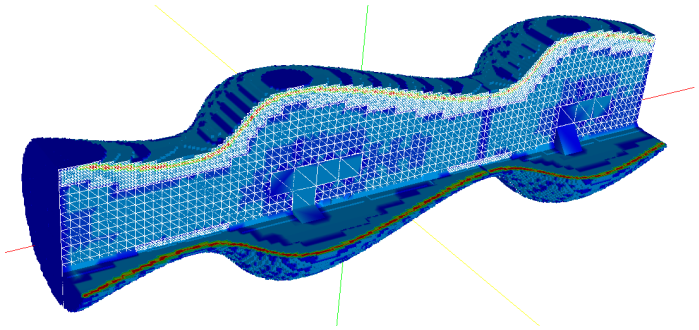
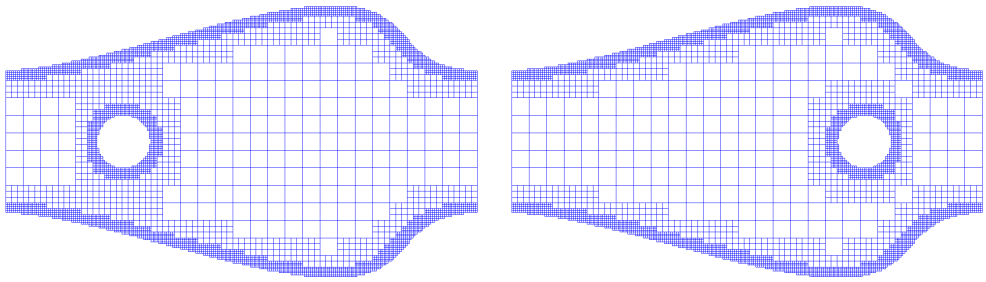


FIG. 4.1. *Three-dimensional adaptive Cartesian grid for two geometric periods of a drift ratchet pore.*

The geometry of the pore is given either in analytical form (see [32] for details) or – like the particles – as an approximated surface triangulation. Based on this geometric information, our Cartesian grids including geometry information are constructed in a recursive manner (see Sect. 5 for details).

The motion of the particles is realised via an Eulerian approach based on the marker-and-cell method that was originally introduced in the sixties by Harlow and Welch for the simulation of free surface problems in [19]. We transfer this method to our moving particles, that is each particle is represented by a certain number of grid cells marked as 'particle' cells. In the case of a regular grid, a change of the particle position only induces a change of those cell markers but leaves the grid itself unchanged. It is obvious that this is a local and, thus, very cheap operation. In the case of an adaptive grid, possible adaptive refinements around the particle (either for a better representation of the particle or for a more accurate discretisation of the velocities and/or the pressure) should 'follow' the particle movement (see Sect. 5.2). As an example, Fig. 4.2 shows a two-dimensional adaptive Cartesian grid for refinement level six at the initial and the final position of a particle. Even this redefinition of the adaptive refinements is a rather local operation which is, in addition, facilitated by the location-aware recursive structure (cf. [18]) of our grids. In general, we stay far below the maximal costs of one sweep over all (old and new) grid cells.

As a consequence, the construction of the modified grid is much cheaper than in the case of Lagrangian approaches using mesh deformation techniques that require the solution of an additional partial differential equation to achieve a numerically useful grid (see [9], e.g.). Thus, the simulation of large displacements of particles which may even cross the complete computational domain is possible.



(a) (b)

FIG. 4.2. *Visualisation of a single particle in a 2D drift ratchet pore with one chamber: Complete geometry at the initial position (a) and at the final position (b). At this level six of refinement,* 4057 *cells and* 3316 *vertices are used.*

4.2. Finite volumes and finite elements. We use different spatial discretisation schemes for regular and adaptive Cartesian grids: For regular grids, we use a finite volume discretisation that was developed in [12] following [37]. For adaptive grids, we use a combination of finite volumes and finite elements as the generalisation of finite volume discretisations to adaptive Cartesian grids is much more complicated and much less natural at least for operators associated to vertex-centered degrees of freedom.

4.2.1. The Navier-Stokes equations: divergence and pressure gradient. The discretisation of the continuous Navier-Stokes equations (3.1) and (3.2) by finite volumes and finite elements, resp., results in a semi-discrete system of equations:

$$M\mathbf{u}_h = 0 \tag{4.1}$$

$$A\frac{d\mathbf{u}_h}{dt} + D\mathbf{u}_h + C(\mathbf{u}_h)\mathbf{u}_h - M^T p_h = 0. \tag{4.2}$$

Here, A denotes the mass matrix (which may be lumped), D the discrete diffusion and $C(\mathbf{u}_h)$ the discrete convection terms. The discrete gradient M^T is the transpose of the discrete divergence operator M. This results from the fact that the pressure acts as a Lagrange multiplier ensuring the fulfillment of the continuity equation in this context (cf. [37]). For the divergence operator M, we use a finite volume discretisation both in the regular and in the adaptive case. The control volumes in this case correspond to the grid cells. We use (bi)linear interpolation of the velocities at the cell faces to approximate the flux over the faces. As the resulting discrete divergence operator only requires data owned by the cell, that is velocity values associated to the cell's vertices, it can be used for adaptive grids as well as for regular grids.

4.2.2. Consistent forces. Since the force values at the geometric interface carry the coupling information from the fluid to the particles, a very accurate computation of the forces is of utmost importance. Therefore, we use the method of consistent forces

described in [17] which has been adapted to both our finite volume and finite element discretisations (see [5]). Hereby, we just have to use the semi-discrete momentum equations (4.2) at the particle boundary nodes i, $i = 1, \ldots, N$:

$$f_i = \left(A\frac{d\mathbf{u}_h}{dt} + D\mathbf{u}_h + C(\mathbf{u}_h)\mathbf{u}_h - M^T p_h \right)_i .$$

The occurring operators are evaluated already for the flow simulation analogue to the operator evaluation at inner fluid nodes with the only difference that the corresponding integrations are performed over fluid cells only and not particle cells that touch the particle boundary nodes. As particles are considered as rigid bodies, we do not need local forces and, in paricular, do not have to map forces to the boundary nodes of a structure solver grid, but only have to compute the overall forces acting on the whole particle. These forces result from the local values f_i by simple summation over boundary nodes:

$$F_h = \sum_{i=1}^{N} f_i,$$
$$M_h = \sum_{i=1}^{N} r_i \times f_i,$$

where r_i denotes the radius vector (from the particleS midpoint) of the ith boundary node[1]. These formulas allow for both an accurate and efficient computation of the forces at the interface.

4.2.3. Finite volumes: diffusion and convection. We use the finite volume discretisation for regular Cartesian grids. The discretisation is described in detail in [4]. In this description, we we restrict ourselves the most important features of the two-dimensional Navier-Stokes equations: The control volumes used have the same size as the cells itself but are centred around the vertices of the cells (see Fig. 4.3 for the two-dimensional case). For the diffusion term, we get the well-known five-point stencil in 2D or seven-point stencil in 3D, respectively.

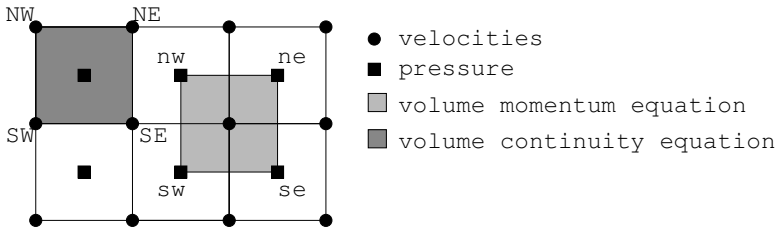


FIG. 4.3. *Allocation of degrees of freedom for velocities and pressure and control volumes for the continuity and the momentum equation in the two-dimensional case (cf. [4]).*

For the convective term, we interpolate the velocity values to the vertices of the control volumes according to a particular interpolation method interconnecting all

[1]There is no scaling with the elements size appearing in the formulas above since this is already done automatically in our finite volume or finite element discretisation, respectively.

components of the three-dimensional velocity vector (see [4, 11] for details). The interpolation is chosen such that the fulfillment of the discrete continuity equation with the divergence operator M from (4.1) is equivalent to the pointwise fulfillment of the continuous form (3.1). We use these interpolated values to approximate the convective flow over the volume faces. Due to the symmetry of the interpolation, the resulting convection operator is antisymmetric [11]. The consistency of the convection operator is guaranteed by the fact that, due to the special interpolation, the continuity equation holds for all four subquadrants of the control volume if it holds for all four cells that share a subquadrant with the control volume [4].

4.2.4. Finite elements: diffusion and convection. In order to achieve the spatial discretisation of the diffusion and the convection operator in the continuous momentum equations (3.2) via finite elements, we use tri- or bilinear ansatz- and test-functions. To tackle locally different refinement levels in our adaptive grids, we use a strictly cell-wise operator evaluation. That is, we compute the corresponding integrals in the weak form of the momentum equations not over the whole supports of the test-functions but, instead, in a cell-wise manner. Thus, the complete operator values for regular grids result from summing up the contributions of (integrals over) all eight/four adjacent cells of the current node. For adaptively refined grids, we have to use suitable interpolation and restriction operators to correctly transport cell-contributions to vertices with differing refinement levels of the adjacent cells (for details, see [18]).

4.3. Time discretisation. Although the adding of transported particles to the fluid flow hardly produces any extra cost or effort regarding the spatial discretisation, we will see in this section that this is not the case for the time discretisation, where we have to take into account several additional aspects. First, we have to handle the (highly) different time scales of the oscillations of the boundary conditions being slow compared to the reaction of the flow field on changes of boundary conditions (cf. Sect. 4.3.2). This first complication only results from the very low Reynolds number and not from the particle transport. However, second, we have to find methods ensuring mass conservation even after a movement of a particle or changed velocities at the particles' surface as a result of solving the equations for the particle motion (cf. Sect. 4.3.3).

4.3.1. Chorin's projection. The semi-discrete Navier-Stokes equations (4.1), (4.2) resulting from the spatial discretisation have to be discretised in time. We realise this by using an explicit Euler step, e.g., in combination with Chorin's projection method [8]:

$$\tilde{\mathbf{u}}_h^{(n+1)} = \mathbf{u}_h^{(n)} + dtA^{-1}\left(D\mathbf{u}_h + C(\mathbf{u}_h)\mathbf{u}_h\right), \tag{4.3}$$

$$(MA^{-1}M^T)p_h^{(n)} = \frac{1}{dt}M\tilde{\mathbf{u}}_h^{(n+1)}, \tag{4.4}$$

$$\mathbf{u}_h^{(n+1)} = \tilde{\mathbf{u}}_h^{(n+1)} - dtM^T p_h^{(n)}, \tag{4.5}$$

where the superscript (n) denotes values associated to time step n or time $t^{(n)}$, respectively. Thus, our time steps are very cheap and, in particular, do not require the solution of a non-linear system of equations as occurring in implicit time discretisa-

tions of the Navier-Stokes equations as a result of the non-linear convection terms $(\mathbf{u} \cdot \nabla)\,\mathbf{u}$.

4.3.2. Decoupling of time scales. For stability reasons, the time steps of explicit schemes are restricted depending especially on the minimum mesh size $h_{\min}$ of the grid. For the explicit Euler method in two dimension, e.g., the corresponding conditions (see [34] e.g.) are

$$dt < \max\left\{\frac{Re}{4}h_{\min}^2, \frac{h_{\min}}{|u_{1,\max}|}, \frac{h_{\min}}{|u_{2,\max}|}\right\}, \tag{4.6}$$

where $|u_{i,\max}|$ is the maximum absolute value of the velocity component i in the computational domain. For a low Reynolds number such as the ones used in our drift ratchet scenario, condition (4.6) reduces to

$$dt < \frac{Re}{4}h_{\min}^2.$$

Thus, we have to reduce our time steps by a factor of four each time we half the spatial mesh width h. In addition, dt becomes very small due to the tiny Reynolds numbers ($Re \leq 0.1$). In order to avoid a huge number of time steps to be computed, we exploit the different time scales of the relaxation of a low Reynolds number flow to a stationary flow as a reaction of varying boundary conditions on the one hand and of the pressure oscillation on the other hand:

- We choose a time step DT approximating the pressure oscillation period with a suitable accuracy:

$$T^{(N)} = N \cdot DT, \quad N = 1, 2, \ldots, \frac{T^{\text{end}}}{DT}.$$

- We compute the stationary flow for the pressure boundary conditions $p_{\text{bound}}^{(N)}$ associated to the current time $T^{(N)}$ by performing small time steps dt fulfilling (4.7) until we reach an equilibrium state. As initial conditions, we use the already known solution at time $T^{(N-1)}$. $\mathbf{u}_h^{(N)}$, and $p_h^{(N)}$ are set equal to the computed equilibrium values.

For an example run (for detailed simulation parameters see Sect. 6.1), we perform small times steps dt until we reach an equilibrium or steady-state solution with zero forces (see Fig. 4.4). The large time steps DT are applied to discretise the changes of the boundary conditions and to update the particle position and (if necessary) its representation in the computational grid.

4.3.3. Particle motion. To capture the motion of a particle, we perform a weak (explicit) coupling between flow solver and particle motion: After each (large) time step DT of the flow solver, we update the particle positions and continue with the next (large) time step of the flow solver. For each update of the particle position, we integrate the motion of the particle's barycentre x_{bary} for the large time steps DT corresponding to the equations (3.3) and (3.4) of Sect. 3 using the forces computed according to Sect. 4.2.2. We currently use an explicit Euler method to perform this numerical integration:

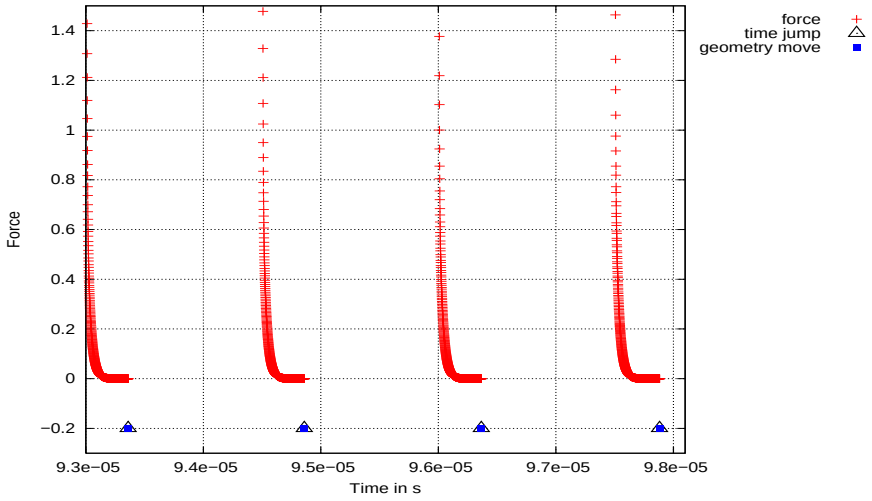


FIG. 4.4. *Forces acting on a particle simulated using two different time scales: small time steps (indicated by crosses) used to compute an equilibrium state (zero forces) and large time steps (indicated by triangles) used to change boundary conditions and to update the particle position if necessary.*

$$u_{\mathrm{p}} = u_{\mathrm{p}} + DT \cdot \frac{F_h}{m_h},$$
$$\omega = \omega + DT \cdot \frac{M_h}{\theta_h},$$
$$x = x + DT \cdot u_{\mathrm{p}}$$

with x denoting the particle position. The rotation of the particle can be neglected in the last equation due to the symmetry of the particles' spherical shape. m_h and θ_h are the discrete approximations of the particle's mass and rotational inertia:

$$m_h = \rho \cdot V_h,$$
$$\theta_h = \sum_{k=1}^{K} m_k {r_k}^2,$$

if ρ is the constant density, V_h the volume of the discrete (grid) representation of the particle, k the index of a grid element associated to the inner part of the particle, K the number of elements representing the inner part of the particle, and r_k the radius of the respective element k with respect to the centre of the particle.

In addition to the position update, the boundary conditions of a node i at the particle's surface have to be set to the new velocities

$$u_i = u_{\mathrm{p}} + \omega \times r_i, \tag{4.7}$$

where r_i is the radius vector of the barycentre x_{bary} to the node i.

If the particle movement (possibly accumulated over several time steps) is large enough, the discrete representation in the Eulerian Cartesian grid of the flow solver will change. Thus, some grid points "loose" their degrees of freedom, others covered by a particle before become new degrees of freedom or new boundary points.

These changes of boundary conditions or particle positions in the spatial grid generally harm the continuity equation. If the continuity equation is not valid for the current velocity field, we get a wrong pressure (containing an artificial part enforcing mass conservation) and, thus, incorrect forces as boundary conditions for the computation of the particle's movement in the subsequent time steps. In order to tackle this issue, we introduce an intermediate step similar to Chorin's projection that restores the mass conservation: We define an artificial variable q and define the new velocities $\mathbf{u}_h^{\text{new}}$ after the projection step as

$$\mathbf{u}_h^{\text{new}} = \mathbf{u}_h^{\text{old}} - M^T q \quad \text{in the domain,} \tag{4.8}$$

$$\mathbf{u}_h^{\text{new}} = \mathbf{u}_h^{\text{old}} \quad \text{at the boundary of the domain.} \tag{4.9}$$

Thus, to use q to enforce mass conservation, we have to solve

$$(MM^T)q = M\mathbf{u}_h^{\text{old}} \quad \text{in the domain,} \tag{4.10}$$

$$\frac{\partial q}{\partial \vec{n}} = 0 \quad \text{at the boundary of the domain.} \tag{4.11}$$

Note that the system matrix MM^T of equation (4.10) is not equal to the usual pressure Poisson system matrix $MA^{-1}M^T$ of (4.4).

If we only have to handle changes of the local refinement depth of the grid (e.g. as a result of the evaluation of an adaptivity criterion), we do not even have to solve an equation but only to use a suitable interpolation. For this purpose, we use the interpolation methods directly resulting from a basis described in [4].

4.3.4. Algorithm. Our time-discretisation results in the following algorithm (visualised in Fig. 4.5): After an initialisation of the geometry and variables in the computational domain, the global time loop starts. Herein, the global boundary conditions are set (for example oscillating pressure at inlet and outlet) and the new cell marker values due to jumps of the particles are computed. Afterwards, the new velocities at the particle boundary have to be computed according to (4.7). Chorin's projection scheme (4.3)–(4.5) is then used in order to get the new velocity flow field that (eventually) has to be made divergence-free via (4.8)–(4.11). The method of consistent forces of Sect. 4.2.2 is applied to compute the forces on the particle that are used to check for the low Reynolds number steady state. If the flow regime is steady, a large time step DT is carried out. It might cause a jump of the particle over one ore even more Cartesian grid cells. Otherwise, the time t is updated by a small time step dt and the algorithm continues with an update of the particle boundary conditions.

5. The flow solvers: F3F and Peano. We use two different codes for the simulation of the fluid flow in the drift ratchet scenario: F3F and Peano. F3F offers the full functionality required for our application scenarios, whereas Peano is our completely new code validating that different sophisticated numerical techniques work

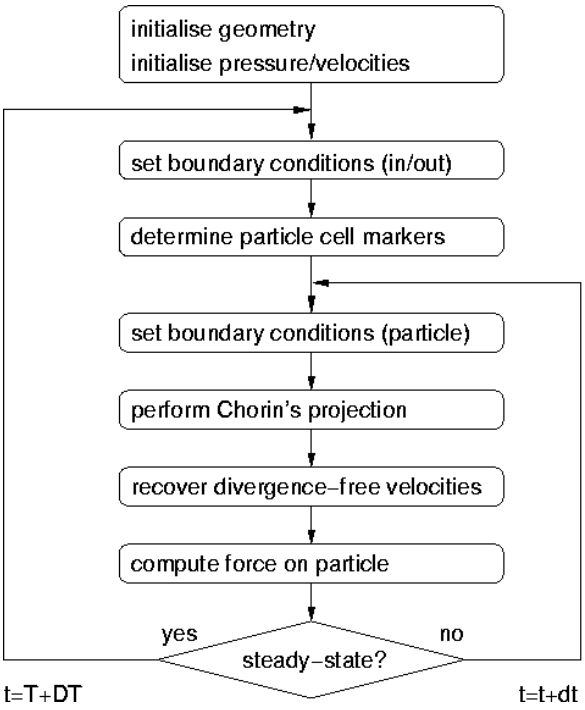


FIG. 4.5. *Schematic representation of the complete particle-flow interaction algorithm. Herein, dt represents the small time step carried out until the low Reynolds number steady state is reached, whereas DT denotes the large time step that is performed once the flow regime is steady.*

hand-in-hand if implemented in a suitable manner. Whereas F3F supports regular three-dimensional grids, Peano is capable of using both two- and three-dimensional adaptive as well as regular grids.

With F3F we solve application problems such as those presented in this paper. F3F is a parallel code capable of simulating all relevant phenomena of our example. Compared to F3F, Peano reduces cost: First on the numerical side by using adaptivity for Cartesian grids (resulting in both lower memory requirements and lower runtimes than for a regular grid with comparable accuracy); second, on the implementational side by the introduction of sophisticated data structures based on space-filling curves (resulting in a high cache efficiency). Peano shall be used for future simulations as soon as all features are completely integrated. Only a two-dimensional Navier-Stokes solver not yet capable of moving geometries is available.

Hence, we have both a stable and well-engineered code for actual computations and a second code which is tailored to numerical (adaptive grids, multigrid) and implementational (memory, parallelisation) efficiency, that will enable us to cope with the future challenges stemming from enhanced models (Brownian motion, etc.), to compute larger scenarios, and in which we are able to integrate a more sophisticated time stepping scheme.

5.1. F3F. F3F has a well-structured modular organisation, being composed of different modules providing well-defined functionalities and interfaces. The regularity

of the grid allows for the usage of simple data structures and algorithms. For FSI scenarios, F3F can completely handle the forces exerted by the flow on some rigid body, as well as the motion of that body in the flow. Furthermore, for more complex situations, the program provides an interface to FSI❄ce for FSI simulations in a partitioned approach. For our particular application, we additionally implemented the algorithm from 4.3.4 to efficiently handle the different time scales.

The simulation program is used in a dual mode: serial and parallel. Due to the simplicity of the underlying data structures, the program uses a static domain decomposition for parallelisation. The partitioning of the computational domain is performed once in the setup phase of the program resulting in a static load balancing. Since F3F does not support adaptivity of the discretisation grid, the computational demands do not vary during the simulation. Thus, there is no need for complicated dynamic load balancing.

Another characteristic of F3F is the explicit assembly of the pressure equation system (4.4). This enables the easy integration of modules and use of different solvers for systems of linear equations, highly tuned for various computer architectures. Usually, we use our own implementation of a preconditioned Conjugate Gradient algorithm. By providing for this task a separate module, F3F allows for simultaneously different parallelisation schemes for the computational domain and for the solver of the linear equations, respectively. Especially for long runs, checkpointing of the simulation is implemented.

Thus, F3F can flexibly tackle various application scenarios, ranging from flows in simple geometries, to rigid bodies swimming in the fluid; short or long simulation runs can be computed in a serial or a parallel mode.

5.2. Peano. In contrast to F3F, the fluid solver Peano is not based upon an a priori discretisation of the computational domain. Instead, the fluid grid is created throughout the computation. Hereby, we embed the computational domain, i.e. the fluid domain where the Navier-Stokes equations are to be solved, into the unit square or unit cube depending on the problem's dimension $d \in \{2, 3\}$. This geometric primitive is then subdivided into three pieces along each coordinate axis. For the resulting smaller subelements we repeat the subdivision recursively. The refinement process stops depending on the numerical accuracy to be achieved, the smoothness of the solution, and the boundary of the computational domain. This method corresponds to an octree approach [26], where the bi-partitioning is replaced by tri-partitioning, and yields an adaptive Cartesian grid. We will give a reason for the tri-partitioning in the following. The refinement process returns a grid hierarchy, where each refinement step corresponds to one grid level. Furthermore, not only the refinement process but also the resulting grids can be interpreted as a tree. We exploited this tree structure for the features implemented wherever possible, and all of these were implemented within a depth-first tree traversal.

One outstanding property of the fluid solver is the complete lack of the need to set up any global matrix. Thus, the memory required for the solution of the Navier-Stokes equations equals the memory needed to represent the data on the grid, and this amount is rather low. No additional (sparse) matrix data is needed. To be able to solve the arising systems of equations, we traverse the grid in an element-wise manner.

Throughout the traversal, the solver's update scheme is applied directly on the grid's vertices. Thus, every step of an iterative solver corresponds to one grid traversal.

In each time step, we use the grid of the predecessor time step as a starting point. During the grid depth-first traversal, one bit per tree node is sufficient to indicate whether this element is refined or not. Whenever an unrefined element, a leaf, is identified that should be refined (as the domain's boundary runs through the element, e.g.) the bit is set, the 3^d subelements are created and the traversal continues. The coarsening is realised in an analogous manner the other way round. To decide whether an element is to be refined, we use standard error estimators ([22], e.g.) throughout the pressure calculation. Furthermore, at the beginning of a time step, we identify top-down all the elements the boundary is running through, and update them where necessary. Thus, a moving boundary triggers the update of a small (local) number of elements only, and the grid's boundary refinement but not the grid move with the domain's changing boundary.

As the grid construction process yields a grid hierarchy, it is preferable to use a geometric multigrid approach to solve the pressure equations. Again, this perfectly fits to the top-down traversal: During the top-down steps we realise the interpolation, and during the bottom-up steps we implement the restriction. As a complete depth-first traversal would be a waste of time for most multigrid smoothing steps, we cut the tree horizontally below the level the algorithm is smoothing at that moment. As we interpolate the solution during the top-down traversal steps, the handling of hanging nodes is rather simple: On hanging nodes no degrees of freedom are located but the solution of the coarser grid is interpolated. Thus, we end up with a consistent, smooth discrete solution [40].

The efficiency of both the dynamic adaptivity and the multigrid algorithm relies on the performance one is able to achieve when traversing the discrete grid. Thus, we combined three different components to realise the algorithm:

- First, we use a space-filling curve to derive the order in which the grid is traversed. The other two components are based upon this technical fundament.
- Second, we do not use any pointer data structure to store the elements, the vertices and the associations between them but use only stacks [18]. Stack containers imply a high spatial and temporal locality for the data access. Hence, the cache-hit rates of the resulting algorithm are very high. We measured rates above 98 percent for all run examples 98. High cache-hit rates in turn are an important ingredient to end up with a high-performance application [10], but not automatically given for PDE solvers. In [41], the hitrates for data to be found in the L2-cache or higher levels of the memory hierarchy are measured as 83 percent for a standard implementation of a red-black Gauss-Seidel solver on a 1024×1024 grid and 97.1 percent for a cache-optimzed cersion. Furthermore, we thus avoid the overhead implied by pointer data structures.
- Third, we parallelised the traversal cutting the space-filling curve into equal-sized pieces, ending up with a domain decomposition approach. The resulting subdomains are known to be quasi-optimal [6], and there are many load-balancing approaches benefiting from space-filling curves [24].

6. Particle transport in the drift ratchet. To examine the movement of a single particle in one pore of the drift ratchet, we performed several simulation runs with different scenarios. Here, we still neglect the Brownian motion in order to isolate effects induced by the geometric asymmetry and the pressure oscillation from those stemming from the interactions with the Brownian motion.

6.1. Three-dimensional simulations with F3F. In all scenarios described below, we used a pore geometry with two chambers similar to the setting displayed in Fig. 6.1. The parameters used in our scaled simulation scenario are: density $\rho = 10^3\,\mathrm{kg/m^3}$, dynamic viscosity $\eta = 10^{-3}\,\mathrm{Pa\,s}$ (fluid), mean velocity at the inflow $u_{\mathrm{mean}} = 0.1\mathrm{m/s}$. In addition, the characteristic length (minimum diameter of the pore) and the particle diameter were set to $1\,\mu$m and $0.6\,\mu$m, respectively. These properties lead to a flow regime with $Re = 0.1$.

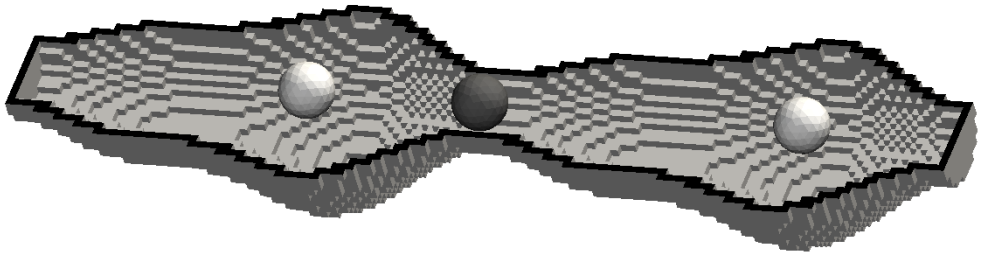


FIG. 6.1. *Three-dimensional asymmetric pore with two chambers. The particle is displayed at three different time steps (i.e. different locations) and the color of the particle corresponds to its velocity (light color – low, dark color – high).*

6.2. Simulation results. In the first scenario, shown in Fig. 6.1, we computed the particle transport through a pore with constant velocity boundary conditions. The specified mean flow velocity was $u_{\mathrm{mean}} = 0.1\mathrm{m/s}$. Thus, the particle moves through the pore from the right to the left.

Fig. 6.2 shows the particle's velocity in axial direction during its way through the pore. The development of the velocity follows the change of the flow velocity induced by the varying diameter of the pore very well. To be more precise, the velocity of the particle in the narrowed region between the two chambers reaches a maximum of $0.16\,\mathrm{m/s}$. For very small (diameter almost zero) particles, we would get a maximum velocity of $0.20\,\mathrm{m/s}$ in the same region. This explicitly shows the interaction of the particle with the flow field. Thus, it also underlines the need of a fully modeled particle movement in such situations. Hence, for the considered case it is not sufficient to use models of massless particles with passive behaviour.

The second scenario is of more practical interest as it already includes oscillating boundary conditions. Therefore, we used pressure boundary conditions where we set the pressure on the left-hand side of the pore to zero, while the pressure on the right-hand side oscillated within the range of $p_{\mathrm{min}} = -11\,\mathrm{kPa}$ and $p_{\mathrm{max}} = 11\,\mathrm{kPa}$. The frequency f was set to $7\,\mathrm{kHz}$. Figs. 6.3 and 6.4 show the slightly delayed reaction of the particle's velocity and movement to the induced oscillating flow field. Clearly, a directed movement of the particle cannot be observed yet due to the small number of simulated periods.

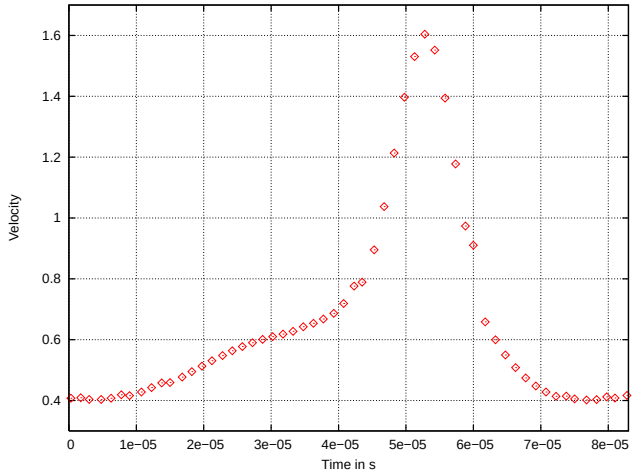


FIG. 6.2. *Velocity of a particle in an asymmetric pore, using a constant inflow condition. The velocity is given in units of* $u_{\text{mean}} = 0.1\,\text{m/s}$. *The simulation was performed on a grid with mesh size* $h = 0.833\ \mu\text{m}$.

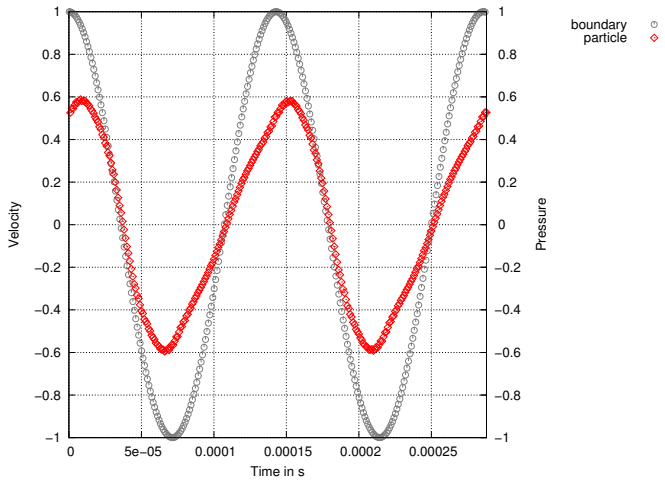


FIG. 6.3. *Velocity of a particle in an asymmetric pore, using an oscillating boundary condition* ($f = 7$ kHz). *The pressure on the right-hand side of the pore and the velocity of the particle are given in units of the amplitude and of* $u_{\text{mean}} = 0.1\,\text{m/s}$, *respectively.*

In the third scenario, we again used the oscillating boundary conditions, but we doubled the frequency of the driving pressure. Fig. 6.5 shows the resulting oscillations of the particle's position. In addition to the frequency of the induced flow, we observe a second, lower frequency movement of the particle. The fact that we discovered low frequency components in the particle movement lead us to the idea that a whole series of frequencies for the particle movement could exist. If this series of frequencies tended to zero, it would, in the end, lead to a pure displacement and hence to a drift of the particle. However, very low frequencies can only be observed for very long simulation times which go beyond the scope of this paper.

These results emphasise the necessity of using fully modeled particle movements and

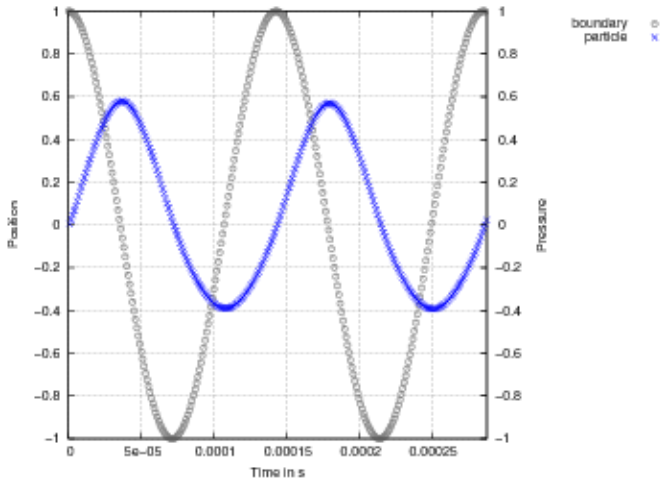


FIG. 6.4. *Relative position of a particle in an asymmetric pore, using an oscillating boundary condition ($f = 7$ kHz). The pressure on the right-hand side of the pore is given in units of the amplitude, and the position is relative to the center of the pore.*

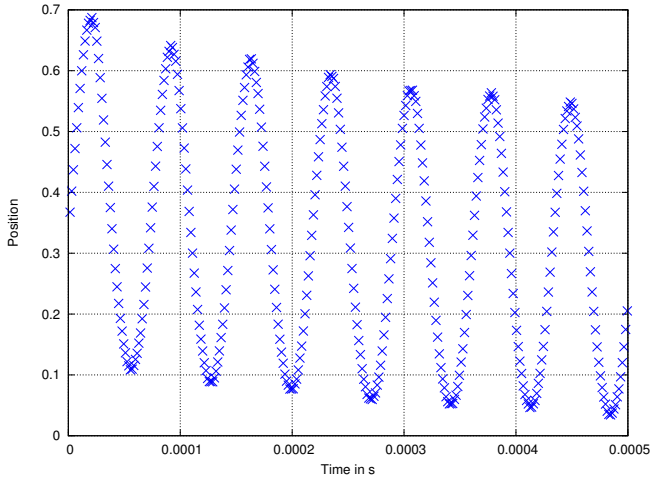


FIG. 6.5. *Relative position of a particle in an asymmetric pore, using an oscillating boundary condition ($f = 14$ kHz). The position is relative to the center of the pore, as in Fig. 6.4.*

show the correct functioning of our FSI approach for the motion of particles in low Reynolds number flows through asymmetric geometries.

6.3. Performance of the time discretisation. Table 6.1 presents the average number of time steps of size dt needed for simulating a time advance $DT = 1.5\,\mu$s for different resolutions of the computational grid. Due to the condition (4.6), dt has to be reduced with decreasing mesh size h as shown in the first two columns of the table. The third column of Table 6.1 holds the number of small time steps dt theoretically necessary for DT, whereas the last column shows the average number of time steps of size dt that have effectively been performed according to our decoupling approach for the time scales (cf. Sect. 4.3).

TABLE 6.1

Theoretically necessary and effectively computed number of small time steps of size dt, according to the decoupling approach (cf. Sect. 4.3), required for a time advance $DT = 1.5$ μs. The results correspond to different mesh sizes h at a frequency of $f = 10$ kHz.

h (μm)	dt (ns)	time steps (theor.)	time steps (eff.)
0.0833	5.79e-1	2,590	412
0.0625	3.25e-1	4,615	218
0.0416	1.45e-1	10,345	314
0.0333	9.26e-2	16,199	1,238

Depending on the concrete resolution level, we needed to compute only about 3–16 percent of the theoretically necessary number of time steps. These results point out the advantage of employing the two time scales: progress of the simulation time with lower computational costs.

6.4. Two-dimensional computations with Peano. This section describes flow results obtained with Peano for two-dimensional flow scenarios simulated on adaptivly refined grids. We have choosen the benchmark "laminar flow around a cylinder" [35] to compare the results of different grid setups with hard reference data. The test case scenario 2D–1 represents a steady-state flow around a cylinder placed in the front part of a two-dimensional channel at Reynolds number 20. The channel has a height and length of 0.41 and 2.46, respectively. The drag and lift coefficients c_d and c_l serve as reference data for the evaluation of the simulations' accuracy. The accuracy depends of course both on the resolution of the cylinder (i.e. the maximum grid level of the spacetree) and on the solution accuracy in the whole domain (depending on the discretisation order, the mesh resolution, and the adaptivity pattern)[2]. All simulations have been performed on an Intel Pentium 4 architecture with a single processor of 3.4 GHz, 1 MB level-2 cache, and 2 GB RAM. We used the gcc 4.1.1[3] with aggressive optimisation.

TABLE 6.2

Survey simulation results of the benchmark 2D–1 [35] computed on grids with a steep adaptive refimnement around the cylinder (cf. Figures 6.6(a) and 6.6(b). The maximum and minimum level of refinement represent the finest and coarsest mesh size in use. The drag (c_d) and lift (c_l) coefficients of the cylinder as well as the runtime of one time step in seconds are shown in the last three columns. The reference force coefficients taken from [35] are given in the last line.

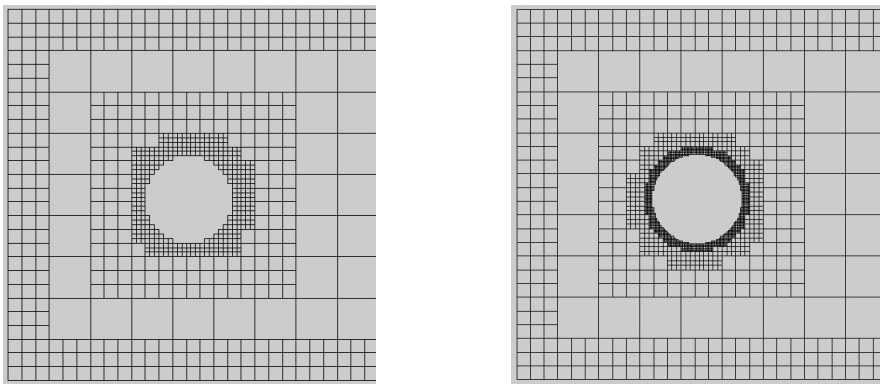
max. level	min. level	cells	vertices	c_d	c_l	CPU/time step [sec]
6	5	1622	1309	5.656	0.0324	0.060
7	5	1953	1578	5.521	0.0220	0.075
8	5	3017	2466	5.569	0.0158	0.100
9	5	5469	4438	5.540	0.0156	0.180
ref. data	-	-	-	5.580	0.0107	-

Table 6.2 shows the results for a suite of scenarios that have all the same minimum spacetree level five corresponding to the coarsest mesh size $h_{\max} = 0.0455$. Around the cylinder, the grid is refined up to a maximum refinement level of nine (i.e.

[2]At the moment, we achieve only first order accuracy at boundaries not conforming with grid lines. The development of a second order boundary discretisation following [3] or [39].

[3]See http://gcc.gnu.org/.

$h_{\min} = 0.00056$). Figures 6.6(a) and 6.6(b) show cut-offs of the corresponding grids for maximum grid levels of seven and eight. The "steep" adaptivity at the cylinder geometry (and there only) results in a quite moderate raise in the number of degrees of freedom. Since the coarsest mesh cells are quite close to the cylinder (see Figure 6.6), the coarse (and, therefore, inaccurate) flow data have a non-negligeable negative impact on the force coefficients, even if the resolution of the cylinder geometry is quite fine. This will be remedied by the usage of a posteriori error estimators for dynamic adaptivity. Apart from accuracy considerations, Table 6.2 also show that the runtimes grow proportionally to the number of degrees of freedom. Thus, the adaptivity pattern and, in particular, the balancedness of the underlying spacetree do not have any influence on the runtime per degree of freedom or, in other word, we do not pay a price for complicated adaptivity patterns in terms of runtime. [4]



(a) (b)

FIG. 6.6. *Visualisation of the adaptive grid refinement on level 7 (a) and 8 (b) of the benchmark flow around a cylinder 2D–1 (cf. [35]). The coarsest resolution in the channel domain is of level 5.*

TABLE 6.3

Survey of adaptive simulation results of the benchmark 2D–1 [35] with smaller coarsest mesh size (i.e. higher minimum spacetree refinement level). The drag (c_d) and lift (c_l) coefficients of the cylinder as well as the runtime of one time step in seconds are shown in the last three columns. The reference force coefficients taken from [35] are given in the last line.

max. level	min. level	cells	vertices	c_d	c_l	CPU/time step [sec]
6	6	4342	4167	5.678	0.0443	0.08
7	7	39073	38494	5.558	0.0135	0.69
8	7	39973	39222	5.684	0.0147	0.71
9	7	42501	41270	5.591	0.0113	0.78
ref. data	-	-	-	5.580	0.0107	-

We have run an additional sequence of simulations, shown in Table 6.3, in order to observe the effect of a better solution due to a more regular refinement in the overall domain. The first two scenarios with level six and seven, respectively, represent a completely resolved Cartesian grid (treated with the adaptive algorithm). The second two scenarios with maximum spacetree refinement level eight and nine, respectively, both keep the minimum level of seven and use additional adaptive refinement at the

[4] For comparisons of runtimes achieved with our methods compared to runtime-optimised solvers on regular grids see [22].

cylinder. We observe a good convergence of the force coefficient towards the reference values. As expected, a further adaptive refinement at the cylinder results in a higher accuracy of the forces if the difference between the maximum and minimum refinement level is not too large (compare level nine in Tables 6.2 and 6.3). To get a feeling for the benefit of grid adaptivity, we compare setups of Tables 6.2 and 6.3 with similar accuracies of the force coefficients: Lines one and two in Table 6.3 (i.e. level six and seven) correspond to lines one and three in Table 6.2 (i.e. maximum level six and eight). The number of required degrees of freedom reduces by a factor of about three and fifteen, respectively, in the case of real adaptivity.

Concerning the memory requirement of the adaptive grid, Table 6.4 shows the amount of bytes needed for one cell and one vertex, respectively, compared to the regular grid holding only fluid degrees of freedom. The administration of grid adaptivity demands, thus, only twelve bytes per grid cells and about four bytes per grid vertex.

TABLE 6.4
Memory required for the cells and vertices on regular and adaptive Cartesian grids in Peano.

	bytes per cell	bytes per vertex
regular grid	24	96
adaptive grid	36	100

7. Conclusion. In this paper, we presented a model, discretisation schemes, and two simulation codes for the transport of particles in a complex geometry on the micro scale. The underlying flow field is characterised by very small Reynolds numbers ($Re \leq 0.1$). The results obtained show that our numerical and implementational approaches are suitable to meet the requirements underlying this application. In contrast to and beyond established approaches for the simulation of suspended particles in fluid flows, an important focus of our work is on the correct computation of the forces acting on the particles. This is essential for our drift ratchet application to achieve a sufficient accuracy even in critical situations such as large particles in narrow regions (between two chambers of a pore, e.g.). Moreover, the same approach is well suited for more general fluid-structure interaction problems as well, where the correct force computation is of utmost importance. Thus, the presented work is complete in the sense that we could show the effectiveness of our methods for the example of the drift ratchet application.

On the other hand, we consider it as a preliminary study for further simulations of the drift ratchet scenario on the basis of models enhanced by further effects such as Brownian motion and particle interactions. Nevertheless, the current simulation results already give hints on a certain directed transport of particles which makes these further examinations very promising. For the simulation of more general fluid-structure interaction problems, the present state of our work marks the point from which we can now start to perform partitioned simulations for a great variety of scenarios and to test different possibilities for components such as interpolation and projection operators between the non-matching solver grids and the coupling strategy. Actually, these two aspects – more detailed simulations for the drift ratchet and enhancement of the methods for general fluid-structure interactions – will be the main focus of our future work.

Acknowledgements

This work was supported by the *German Research Foundation*, project HA 1517/25-1 and HA 1517/25-2 and research group 493. This support is thankfully acknowledged.

Furthermore, we thank our project partners P. Hänggi, P. Talkner, and M. Kostur (Theoretical Physics I, Universität Augsburg) for their contributions on the experimental side.

REFERENCES

[1] R. D. Astumian and P. Hänggi. Brownian motors. *Phys. Today*, 55:33–39, 2002.
[2] M. Bader, A. Frank, and C. Zenger. An octree-based approach for fast elliptic solvers. *High Performance Scientific and Engineering Computing*, 21:157–166, 2002.
[3] M. Bozkurttas, H. Dong., V. Seshadri, R. Mittal, and F. Najjar. Towards numerical simulation of flapping foils on fixed cartesian grids. In *43rd Aerospace Sciences Meeting and Exhibit*, 2005. AIAA 2005-0079.
[4] M. Brenk, H.-J. Bungartz, M. Mehl, and T. Neckel. Fluid-structure interaction on cartesian grids: Flow simulation and coupling environment. In H.-J. Bungartz and M. Schäfer, editors, *Fluid-Structure Interaction*, number 53 in LNCSE, pages 233–269. Springer, 2006.
[5] M. Brenk, H.-J. Bungartz, and T. Neckel. Cartesian discretisations for fluid-structure interaction – consistent forces. In P. Wesseling, E. Oñate, and J. Périaux, editors, *ECCOMAS CFD 2006, European Conference on Computational Fluid Dynamics*. TU Delft, 2006.
[6] H.-J. Bungartz, M. Mehl, and T. Weinzierl. A parallel adaptive Cartesian PDE solver using space–filling curves. In E. W. Nagel, V. W. Walter, and W. Lehner, editors, *Euro-Par 2006, Parallel Processing, 12th International Euro-Par Conference*, volume 4128 of *LNCS*, pages 1064–1074, Berlin Heidelberg, 2006. Springer-Verlag.
[7] S. Canic and A. Mikelic. Effective equations modeling the flow of a viscous incompressible fluid through a long elastic tube arising in the study of blood flow through small arteries. *SIAM J. Appl. Dyn. Syst.*, 2:431–463, 2003.
[8] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22:745–762, 1968.
[9] Y. Di, R. Li, T. Tang, and P. Zhang. Moving mesh finite element methods for the incompressible Navier–Stokes equations. *SIAM J. Sci. Comput.*, 26(3):1036–1056, 2005.
[10] C. C. Douglas, J. Hu, M. Kowarschik, U. Rüde, and C. Weiss. Cache optimization for structured and unstructured grid multigrid. *Electron. T. Numer. Ana.*, 10:21–40, 2000.
[11] M. Emans. *Numerische Simulation des unterkühlten Blasensiedens in turbulenter Strömung: Ein Euler-Lagrange-Verfahren auf orthogonalen Gittern*. PhD thesis, Institut für Informatik, TU München, 2003.
[12] M. Emans and Ch. Zenger. An efficient method for the prediction of the motion of individual bubbles. *Int. J. Comp. Fluid Dyn.*, 19:347–356, 2005.
[13] H. Faxén. Die Bewegung einer starren Kugel längs der Achse eines mit zäher Flüssigkeit gefüllten Rohres. *Ark. Mat. Astr. Fys.*, 17(27), 1923.
[14] Fraunhofer SCAI, Sankt Augustin. *MpCCI 3.0: Multidisciplinary Simulations through Code Coupling*, 2007. Manual.
[15] Y. C. Gerstenmaier, A. Castellazzi, and G. Wachutka. Electrothermal simulation of multichip-modules with novel transient thermal model and time-dependent boundary conditions. *IEEE T. Power Electr.*, 21:45–55, 2006.
[16] E. Givelberg and K. Yelick. Distributed immersed boundary simulation in titanium. *SIAM J. Sci. Comput.*, 28:1361–1378, 2006.
[17] P. M. Gresho and R. L. Sani. *Incompressible Flow and the Finite Element Method*. John Wiley & Sons, 1998.
[18] F. Günther, M. Mehl, M. Pögl, and C. Zenger. A cache-aware algorithm for PDEs on hierarchical data structures based on space-filling curves. *SIAM J. Sci. Comput.*, 28(5):1634–1650, 2004.
[19] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *Physics of fluids*, 8(12):2182–2189, 1965.
[20] C. Kettner, P. Reimann, P. Hänggi, and F. Müller. Drift ratchet. *Phys. Rev. E*, 61:312–323, 2000.

[21] S. Matthias and F. Müller. Asymmetric pores in a silicon membrane acting as massively parallel brownian ratchets. *letters to nature*, 424:53–57, 2003.

[22] M. Mehl, T. Weinzierl, and C. Zenger. A cache-oblivious self-adaptive full multigrid method. *Numer. Linear Algebr.*, 13(2-3):275–291, 2006.

[23] T. F. Miller, M. Eleftheriou, P. Pattniak, A. Ndirango, D. Newns, and G. J. Martyna. Symplectic quaternion scheme for biophysical molecular dynamics. *J. Comput. Chem.*, 116(20):8649–8659, 2002.

[24] W. F. Mitchell. A refinement-tree based partitioning method for dynamic load balancing with adaptively refined grids. *J. Parallel Distr. Com.*, 67(4):417–429, 2007.

[25] F. Müller, A. Birner, J. Schilling, U. Gösele, C. Kettner, and P. Hänggi. Membranes for micropumps from macroporous silicon. *Phys. Status Solidi A*, 182:585–590, 2000.

[26] R.-P. Mundani. *Hierarchische Geometriemodelle zur Einbettung verteilter Simulationsaufgaben*. Berichte aus der Informatik. Shaker Verlag, Aachen, 2006. PhD thesis.

[27] K. C. Park and C. A. Felippa. Partitioned analysis of coupled systems. In T. Belytschko and T. J. R. Hughes, editors, *Computational Methods for Transient Analysis*, pages 157–219. Elsevier, 1983.

[28] S. Piperno, C. Farhat, and B. Larrouturou. Partitioned procedures for the transient solution of coupled aeroelastic problems part I: Model problem, theory and two-dimensional application. *Comput. Methods Appl. Mech. Engrg.*, 124:79–112, 1995.

[29] P. Reimann. Brownian motors: noisy transport far from equilibrium. *Phys. Rep.*, 361:57–265, 2002.

[30] P. Reimann and P. Hänggi. Introduction to the physics of brownian motors. *Appl. Phys. A–Mater*, 75:169–178, 2002.

[31] D. Rickwood and B. D. Hames. *Gel Electrophoresis of the Nucleic Acids: A Practical Approach*. Oxford University Press, 1990.

[32] M. Schindler. *Free-Surface Microflows and Particle Transport*. PhD thesis, Institut für Physik, Universität Augsburg, 2006.

[33] P. Serwer and G. A. Griess. Advances in the separation of bacteriophages and related particles. *J. Chromatogr. B*, 722:179–190, 1999.

[34] M. F. Tomé and S. McKee. GENSMAC: A computational marker and cell method for free surface flows in general domains. *J. Comp. Phys.*, 110:171–186, 1994.

[35] S. Turek and M. Schäfer. Benchmark computations of laminar flow around a cylinder. In E. H. Hirschel, editor, *Flow Simulation with High-Performance Computers II*, number 52 in NNFM. Vieweg, 1996.

[36] R. van Zon and J. Schofield. Numerical implementation of the exact dynamics of free rigid bodies. *J. Comput. Phys.*, 225(1):145–164, 2007.

[37] R. W. C. P. Verstappen and A. E. P. Veldman. Symmetry-preserving discretization of turbulent channel flow. *J. Comp. Phys.*, 187:343–368, 2003.

[38] J. Vetillard and P. Destuynder. A noise control problem arising in a flow duct. *SIAM J. Appl. Math.*, 64:1987–2017, 2004.

[39] T. Wagner. Randbehandlung höherer Ordnung für ein cache-optimales Finite-Element-Verfahren auf kartesischen Gittern. Diploma thesis, Institut für Informatik, TU München, 2005.

[40] W. Wang. Special bilinear quadrilateral elements for locally refined finite element grids. *SIAM J. Sci. Comput.*, 22(6):2029–2050, 2001.

[41] Christian Weiß, Wolfgang Karl, Markus Kowarschik, and Ulrich Rüde. Memory characteristics of iterative methods. In *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*, pages 1–31, New York, NY, USA, 1999. ACM.